3D Video Processing Algorithms – Part II

Lucio Azzari  ■  Olli Suominen  ■  Suomeet Sen  ■  Atanas Gotchev  ■  
Done Bugdaici  ■  Gozde Bozdagi Akar
Abstract: This report describes novel algorithms developed to enhance the quality of 3D video. It concentrates on deblocking and sharpening of stereo video, filling in disoccluded holes in depth image – based rendering, error concealment at the receiver side for both stereo and view plus depth. In addition, optimized implementations of transform-domain filtering and bilateral filtering are presented. Deblocking and sharpening of stereo video is achieved by non-local transform-domain collaborative filtering, while disocclusions are handled by non-local mean type of inpainting, proved to be superior to other methods by subjective tests. Two schemes for error concealment of video plus depth, using either complementary frame or complementary motion vectors are presented. For the case of MVC, concealment using complementary motion vectors is studied. Implementations of transform-domain filtering and bilateral filtering at the DSP core of OMAP3 and a GPU by OpenCL are reported.

Keywords: 4D grouping and transform-domain collaborative filtering, bilateral filtering, non-local means inpainting, OpenCL
Executive Summary

We present the second part of 3D video processing methods. In this report we concentrate on deblocking and sharpening of stereo video, filling in disoccluded holes in depth image –based rendering (DIBR), error concealment at the receiver side for both stereo and view plus depth. In addition, we present optimized implementations of transform-domain filtering and bilateral filtering.

Standard block transform-based compression methods often cause blocking artefacts, which have been found particularly annoying and also degrading the overall quality and the perception of depth. Suitable deblocking is required at the receiver side to tackle such artefacts. Current trend in restoration algorithms suggests applying non-local collaborative filtering methods. In the report, such a technique is proposed for deblocking of stereo video compressed by Simulcast or MVC. The algorithm searches for similar patches exhibiting high spatial correlation along temporal dimension and between the two views. The selected patches are grouped into comprehensive 4D structure, which is decorrelated by efficient transforms, namely DCT along spatio-temporal directions and Haar wavelet transform between stereo frames. Artefact suppression is performed through transform-domain thresholding to result in first empirical estimate of the deblocked signal which is then used for a second-stage transform-domain Wiener filtering. Furthermore, an elegant stereo sharpening can be accomplished in the transform domain by alpha-rooting. We demonstrate that the procedure leads to visually pleasant results and is compares favourably against its simplified versions, i.e. sliding DCT filtering and bilateral filtering.

Next, the report addresses the disocclusion filling problem in DIBR. Three state-of-the-art methods based on dept preprocessing (smoothing), diffusion filtering and non-local inpaing are compared. The latter operates by taking a weighted mean of the k most similar patches to fully recover a patch containing disoccluded areas. This approach proves superior in terms of performance and therefore, it is additionally modified. Improvements are suggested for speeding up the search for similar patches and improving the quality by proper prioritization and non-local mean weighting. In order to prove the feasibility of the method, objective and subjective tests have been performed, comparing the technique with the other techniques. The developed technique shows high and consistent ranks over various video contents with reduced computational cost.

Error concealment algorithms for restoring lost frames in V+D and MVC encoded and transmitted videos are presented. For the V+D representation, two schemes are considered, namely concealment using complementary frame and concealment using complementary motion vectors. For the MVC, concealment using complementary motion vectors is studied and experimental results are provided.

The last section considers optimizations done with respect to mobile platforms in use. OMAP 3530 has been employed to test the suitability of running video processing algorithms on limited computational resources. Main algorithm being targeted is the bilateral filtering as it finds applications in both deblocking and depth map restoration. Furthermore, an alternative approach of using OpenCL to control the graphics accelerator is explored, as it can be expected to be available in near-future mobile devices. The combination of the ARM and DSP processors on an OMAP3 device is found to be insufficient for the bilateral filter. While the results on OpenCL are not definitive due to the lack of actual mobile, OpenCL capable devices, the presented tests on a graphics adapter of a limited-power netbook are presented, which look quite positive.

The work on error concealment has been accomplished by METU team while the rest has been accomplished by the TTY team.
# Table of Contents

1 Introduction .......................................................................................................................... 5

2 Deblocking of stereo video ................................................................................................... 6
   2.1 Introduction ................................................................................................................... 6
   2.2 Block matching and collaborating filtering in 4D-transform domain ....................... 7
       2.2.1 Main idea ............................................................................................................. 7
       2.2.2 Problem formulation .......................................................................................... 7
       2.2.3 Grouping ........................................................................................................... 8
       2.2.4 Collaborative filtering and aggregation ........................................................... 10
   2.3 Experimental results .................................................................................................... 12
       2.3.1 Parameter setting .............................................................................................. 12
       2.3.2 Results ............................................................................................................. 13
   2.4 Sharpening of stereo video .......................................................................................... 16
       2.4.1 Alpha-rooting in 4D-transform domain ............................................................. 16
       2.4.2 Experimental results ......................................................................................... 17
   2.5 Conclusions ................................................................................................................. 18

3 Disocclusion filling .............................................................................................................. 23
   3.1 Introduction ................................................................................................................. 23
   3.2 Proposed approach ..................................................................................................... 25
   3.3 Experimental results .................................................................................................... 29
       3.3.1 Objective tests .................................................................................................. 30
       3.3.2 Subjective tests ................................................................................................. 33
   3.4 Conclusions ................................................................................................................. 37

4 Error Concealment ............................................................................................................. 38
   4.1 Concealment of V+D video .......................................................................................... 38
       4.1.1 Concealment Using Complementary Frame ..................................................... 38
       4.1.2 Concealment Using Complementary Motion Vectors ...................................... 44
       4.1.3 Evaluation of the results .................................................................................... 51
   4.2 Concealment for MVC ................................................................................................. 52
       4.2.1 Concealment Using Complementary Motion Vectors for MVC ..................... 52
       4.2.2 Evaluation of the results on stereo concealment .............................................. 61

5 Implementation of video processing algorithms .................................................................. 62
   5.1 Algorithms ................................................................................................................... 62
       5.1.1 Bilateral filtering ................................................................................................. 62
       5.1.2 Sliding Window DCT ........................................................................................ 63
       5.1.3 VBM3D ............................................................................................................. 63
## 5.2 Implementation environment

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.1</td>
<td>OMAP</td>
<td>64</td>
</tr>
<tr>
<td>5.2.2</td>
<td>OpenCL</td>
<td>64</td>
</tr>
</tbody>
</table>

## 5.3 Implementation

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1</td>
<td>Bilateral filter on OMAP</td>
<td>65</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Bilateral and hypothesis filter in OpenCL</td>
<td>66</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Sliding Windowed DCT and VBM3D on OMAP</td>
<td>67</td>
</tr>
</tbody>
</table>

## 5.4 Experimental results

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.1</td>
<td>Sliding Windowed DCT and VBM3D on OMAP</td>
<td>67</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Bilateral filter on OMAP: DSP+ARM</td>
<td>68</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Bilateral and hypothesis filter on OpenCL</td>
<td>68</td>
</tr>
</tbody>
</table>

## 5.5 Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>69</td>
</tr>
</tbody>
</table>

## 6 Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>71</td>
</tr>
</tbody>
</table>
1 Introduction

This deliverable consists of four parts.

The first part introduces a deblocking and sharpening algorithm that exploits the redundant information present in a 3D video (temporal and inter-channel redundancy) in order to deblock sequences compressed by H.264 AVC or MVC for varying quantization parameters (QP).

In the second part, an extension of the DIBR method is presented tackling the problem of disoccluded areas by non-local means inpainting. The method is evaluated against other state-of-the-art methods by objective comparisons and subjective tests.

The third part deals with error concealment at the receiver side for 3D video represented either by V+D or MVC. Three methods are studied and analyzed.

The fourth part deals with implementing core blocks for restoration algorithms (deblocking, denoising, restoration) on current and future mobile platforms utilizing the specialized hardware capabilities available.

The first and second parts are authored by Lucio Azzari and Atanas Gotchev from TTY, the third part is authored by Done Bugdayci and Gozde Bozdagi Akar from METU, and the fourth part is authored by Olli Suominen and Atanas Gotchev.
2 Deblocking of stereo video

2.1 Introduction

Conventional stereo video has been considered as the format of choice for 3D-enabled mobile devices. Suitable coding methods have been developed and further optimized for mobile use [1]. Their performance has been compared by an extensive subjective test [2]. Three compression methods have been considered:

1. H.264/AVC Simulcast (Sim): the left and the right channels are compressed separately with the standard H264/AVC encoder;
2. H.264/AVC Multi-view Video Coding (MVC): one of the channels is encoded independently employing temporal prediction only and the other is encoded by using both temporal and inter-view predictions
3. Mixed Resolution Stereo Coding (MRSC): one of the views is down-sampled in order to reduce the amount of data for encoding. The resulting quality degraded somehow by the decimation is close to the mean of the two channels [3]. At the receiver side, the view with reduced resolution is up-sampled in order to obtain the full resolution.

All the above compression approaches are based on the block transform based H.364 AVC standard, where the lossy compression effect is achieved by quantization of the transform coefficients. For low bitrate cases, imposed by narrow bandwidth channels, the harsh quantization generates blocking artefacts.

The H.264 standard includes an in-loop deblocking filter [4] and [5], aimed at reducing such artifacts. It is based on an in-loop one-dimensional filter with varying strength applied across boundaries of two different blocks in an adaptive manner. However, often this filter is not enough to prevent blocking in case of high compression. For this reason, a number of deblocking methods have been developed in order to improve the video quality at the receiver side.

Yao et al present four one-dimensional fuzzy-filters for deblocking based on the activity in specific areas. The algorithm classifies areas according to the type of local image: weak and strong edges, texture and smooth areas [6].

A simple deblocking algorithm is presented in [7], in which a blocky image is shifted in all possible directions and is JPEG compressed again; then all compressed versions are re-shifted to the original coordinates and an average of them is obtained. In this way, the block artifacts, present in different parts of the images due to the shift, are reduced by the averaging filter. In a similar way, Chen et al. propose averaging performed adaptively taking into account the characteristics of the human visual system [8].

A more sophisticated deblocking method uses the Projection Onto Convex Sets (POCS). In the approach, if two adjacent blocks are highly correlated, then their singular frequency components are assumed similar to the global frequency components computed using both of them. Altunbasak et al. have employed this correlation in order to reduce blocky artifacts [9].

In [10], a collaborative filter is proposed for video denoising. The algorithm selects similar block patches in the spatio-temporal vicinity of the processed block to group and filter then together in a collaborative transform-domain manner. We aim at modifying this technique for the case of deblocking and to apply it to stereo video in order to benefit also from the similarity between patches in different views.
2.2 Block matching and collaborating filtering in 4D-transform domain

2.2.1 Main idea

The idea of collaborative filtering for image denoising or restoration is to collect similar patches in the processed image or image sequence and to process them together. Similarity helps in emphasizing the essential structure of the collected patches while the noise and other contaminations (e.g., compression artefacts) are expected to have probabilistic behavior and to be suppressed by the collaborative filtering.

At a stage of grouping, a search for similar blocks (patches) is done with respect to a reference block, which initializes the similarity search. Such search can be performed either in spatial vicinity (that is, within the same video frame) or in temporal domain (that is, forward and/or backward along successive video frames or in both [10]. Similarity is defined with respect to some norm, most often the \( l^2 \) norm (smaller distance – closer similarity). In [10], the grouping of similar blocks has been done irrespectively of their original position (in the same or in another video frame). However, this proved to be sub-optimal and that a more advantageous grouping is to group patches along temporal direction separately from grouping patches within the same frame [11].

As soon as enough patches are grouped together, they are processed together. This is the stage of collaborative filtering. A simple way is to get the average of all stacked blocks. This would immediately reduce the amount of present noise by a factor of \( N \) for \( N \) stacked patches. A more sophisticated approach is to apply a multi-dimensional transform on the patch grouping. Consider patches combined in a 3-dimensional volume. A 3D transform can be applied on that volume to decorrelate the data. Information content would be concentrated in high-magnitude transform coefficients while the noise (or another random contamination) would be spread across all coefficients. A suitable thresholding would kill the noise in the small coefficients and still would keep the information content being exhibited by the high-magnitude coefficients. After an inverse transform, the jointly processed blocks are returned back to their original places. The scheme depends on over-completeness. The reference block can be moved with small step thus overlapping the previous reference block. Blocks participate in many different similarity structures. Thus, each pixel in estimate several times as it has been participating in different block similarities. This is the aggregation due to over-completeness of the procedure. The result of the scheme is an estimate of the true information signal. It can serve as pilot estimate in a subsequent transform-domain Wiener filtering [10], [11].

There are several crucial issues to make the above scheme well-performing. First, it is the way of proper grouping. Blocks, exhibiting similar type of correlation should be grouped together into higher-dimensional structure. Second, this is the selection of proper decorrelating transform working on the groped data and the way different estimates of a block are combined together.

The purpose of this work is to extend the collaborative filtering for the case of stereo video contaminated by blocky artefacts. It is essential to find the proper grouping for such modality, to test whether processing two channels together is beneficial and to find the proper thresholding parameters leading to suppression of the unwanted blocky artefacts.

2.2.2 Problem formulation

Denote by \( y^l(x,t) \) and \( y^r(x,t) \) the luminance components of left and right channels of stereo video, where \( x = [x_1, x_2] \) is a variable of the spatial coordinates and \( t \) is time variable.
Once the sequence has been compressed the blocky artifacts become visible due to the quantization phase. The blocky version is modeled as the true version contaminated by additive quantization noise:

\[ y_q^c = y^c + c^e \]  

(1)

where \( y_q^c = (L, R) \), \( y_q^c \) is the quantized version of the original \( y^c \) and \( c^e \sim \mathcal{N}(0, \sigma_e^2) \) is i.i.d Gaussian noise. We aim at finding a deblocked estimate \( \hat{y} \), of the true signal \( y \).

### 2.2.3 Grouping

We largely follow the notations suggested in [11]. Starting from the current frame at time \( t_0 \), in one of the two channels (left channel), a reference block \( B^L(x_0, t_0) \) is selected by placing a window of size \( <w \times w> \) with the top-left corner on coordinate \( x_0 \). The purpose of grouping is to find the coordinates of similar blocks \( B^L(x_i, t_0 + i) \) with \( i = [-M \ldots M] \) in previous and successive frames using a block matching algorithm. Block matching is accomplished with respect to distance metric (usually Euclidean distance). Referring to the distance operator as \( d(B^L(x_i, t_i), B^L(x_i, t_i + 1)) \) the best match is defined as follow:

\[
\begin{align*}
    x_{i \pm 1} &= \arg \min_{x_i \in \mathcal{X}_i} \left\{ d(B^L(x_i, t_i), B^L(x_i, t_{i \pm 1})) \right\} \\
\end{align*}
\]

(2)

where \( \mathcal{X}_i \) is a search window centred at \( x_i \) on the frame \( t_i \pm 1 \). Once found the coordinates of the most similar block are added to the trajectory of \( x_0: x_i \in \text{Traj}(x_0) \). The search continues from the found block to next temporal frame.

As soon as enough time frames are processes, and the trajectory is completely defined, all similar patches are collected in a single volume:

\[
\begin{align*}
    V^L(x_0, t_0) &= \left\{ B^L(x_i, t_i) \right\} \\
\end{align*}
\]

(3)

where \( x_i \in \text{Traj}(x_0) \).

At this point it is necessary to extend the procedure to the right channel. The disparity between the left and right views is explored and the most similar block to \( B^L(x_0, t_0) \) in the right frame at time \( t_o \) is defined as \( B^R(x_0 + d_0, t_0) \), where \( d_0 = [\delta, 0] \) represents a disparity shift (a translation along the horizontal direction assuming rectified cameras):

\[
\begin{align*}
    d_0 &= \arg \min_{d \in \mathcal{X}_0} \left\{ d(B^L(x_0, t_0), B^R(x_0 + d, t_0)) \right\} \\
\end{align*}
\]

(4)

where \( \mathcal{X}_0 \) is the search window centred at \( x_0 \).

Once \( d_0 \) and \( B^R(x_0 + d_0, t_0) \) are found, the search for similar block along the right channel continues for frames preceding or succeeding \( t_0 \). Two approaches are possible:
1. The search process goes as for the left channel, i.e. along time (see Figure 1 (a)): taking 
$B^R(x_0 + d_0, t_0)$ as a reference frame.

2. For each single frames $t_i$ of the left channel, the coordinates of $x_i \in Traj(x_0)$ are used to
find the disparity-shifted by $d_i$ frame in the right ones (see Figure 1 (b)).

In our experiments, the first scheme has been chosen. Using this search scheme, it is possible
to follow fast movement of the objects in the sequence, improving in this way the block matching
result. Also, motion vectors are assumed available which can significantly simplify the similarity
search. In contrast to [10], the collection of similar blocks is done only for temporal frames and
not for spatial blocks in the same frame. This is due to the fact that the temporal redundancy is
stronger than the spatial one, and it ensures good quality with reduced complexity.

Figure 1. Different searching methods
The $\ell_2$ distance has been used to compute the distances among the different patches during the block matching phase.

Once tall similar blocks are found and the two volumes $V^L(x_0, t_0)$ and $V^R(x_0, t_0)$ are created, they are grouped in a 4D structure, as follows:

$$G(x_0, t_0) = \{V^L(x_0, t_0), V^R(x_0, d_0, t_0)\}$$

### 2.2.4 Collaborative filtering and aggregation

The groups are filtered in a collaborative manner. Separable 1D transforms are applied along the four dimensions of the group structure. For the volumes, that is DCT (or 3D-DCT on the channel volumes). For the inter-channel dimension, it is Haar transform, taking the average and the difference between corresponding DCT coefficients form the left-channel and right-channel volume. The 4D hybrid transform effectively decorrelate the data and provides extremely sparse representation of the group patches. The sparse representation allows for suppression of artefacts through thresholding. A two-stage thresholding procedure is employed:

As a first step, a hard-thresholding of the transform coefficients is performed. Denote by $U(x_0, t_0)$ the 4-dimensional transform of $G(x_0, t_0)$ (see Eq. (6)), its coefficients are filtered following Eqs. (7) and (8):

$$U(x_0, t_0) = T_{4D}(G(x_0, t_0)) \quad (6)$$

$$\hat{G}(x_0, t_0) = T_4^{-1}_D \Gamma U \quad (7)$$
where:

\[
\Gamma\{U(\chi)\} = \begin{cases} 
U(\chi) & \text{if } |U(\chi)| \geq \alpha \\
0 & \text{if } |U(\chi)| < \alpha 
\end{cases}
\]  

(8)

In Eq. (8) \( \chi \) is a four-dimensional vector of coordinates within the transformed group.

After the group patches have been restored they are returned to their respective places. As the reference block moves one pixel at a time, each pixels is being estimated several times, thus leading to multiple (aggregated) estimates of each pixel which are collected and weighted to get the final estimate of each pixel. After the first stage, one gets a pilot estimate of the deblocked sequence.

At a second stage, the grouping procedure is performed again. A Wiener filter is employed instead of hard thresholding.

\[
\hat{G}(x_0, t_0) = T_{4D} W_{4D} G(x_0, t_0)
\]  

(9)

In Eq. (9) the Wiener filtering represented by \( W \) is performed like shown in Eq. (10).

\[
W = \frac{|U(x_0, t_0)|^2}{|U(x_0, t_0)|^2 + \sigma^2}
\]  

(10)

The Wiener filter uses the transform coefficients of the pilot estimate to filter the group of noisy patches. The aggregation based on multiple estimates of each pixel is performed again. In [11], it is suggested that the weighting coefficients of the two aggregation steps (after thresholding and after Wiener filtering) are selected to be inversely proportional to the number of thresholded coefficients in the first stage and to the Wiener coefficients in the second stage. In our case, in order to safe time, a direct average of all estimates was used as if was found with sufficient quality compared to the weighted combination.

In Eqs. (8) and (10) there are two constants, \( \alpha \) and \( \sigma \), that will be estimated in Section 2.3.1.

In our specific case a 3D-DCT transform has been used to transform singularly \( V^L(x_0, t_0) \) and \( V^R(x_0, t_0) \), and a wavelet Haar transform along the 4th dimension has been used to generate \( U(x_0, t_0) \).

Figure 3. Filtering process
2.3 Experimental results

2.3.1 Parameter setting

According to the observation model, the quantization noise is considered as an i.i.d. Gaussian noise with zero mean and variance $\sigma^2$. The grouping and collaborative filtering approach critically depends on the estimation of $\sigma^2$ as it controls the thresholds in both the hard thresholding and Wiener filtering stages. A proper estimate of $\sigma^2$ related with the amount of quantization is needed.

The strength of the blocky artefacts is determined by the the quantization parameter (QP) used to quantize the DCT coefficients in the H264 encoder. A higher QP generates stronger and more visible blockiness in the resulting video. Another factor related with the amount of blocky artefacts is the targeted bitbudget given in bits-per-pixel (bpp). A relation between the QP and bpp from one hand and $\sigma^2$ from another does exist and has to be quantified. This relationship is found empirically. A number of experiments on several compressed videos have been carried out to find the optimal $\sigma^2$ for different QP and bpp values. The range of QPs has been varied in a dense grid within the range [32, 50] in order to get well exhibited blocking artefacts. The group of test sequences includes five sequences: “Bullinger”, “Butterfly”, “Car”, “Horse” [24], and “Soccer” [2]. The sequences have been compressed by H.264 encoder in simulcast mode for a range of QP values. Each of the sequences has been processed by the deblocking technique seeking for $\sigma^2$ providing highest PSNR of restoration.

A parametric model has been fitted over the $\sigma$ values by minimizing the Mean Absolute Error. Figure 4 shows the dependence of the optimal variance values with respect to the QP and the bpp. Eq. (11) approximates the bi-variate dependence.
\[
\sigma_{opt} = 35.9 + bpp^{0.181} + 0.0384QP^2 - 2.16QP
\]  
(11)

The parameter \( \alpha \) in Eq. (8) is selected as equal to \( 2.7 \cdot \sigma^2 \).

### 2.3.2 Results

In this Section we present results in terms of PSNR of all sequences using the proposed method. The filtering is done with sigma determined according to Eq. (11). The method is compared against other candidate methods for video deblocking. The compared methods include

- The video version of the BM3D algorithm, as presented in [10] with the estimated value of \( \sigma^2 \); the operation runs independently on the two channels and is denoted as `single channel' in the table
- Sliding Windows DCT for deblocking as presented in [12];
- A deblocking algorithm based on a bilateral Gaussian filter [13].

According to Table 1, the proposed algorithm results in the best PSNRs in all cases. There is an average improvement of about \( 0.7 \, dB \) and in some cases up to \( 1.2 \, dB \). Results vary over different contents. It is interesting to explore the difference between the `single channel' method. The use of both stereo channels in the group for filtering clearly brings an improvement due to the efficient Haar transform operating across this dimension. Note that as this a deblocking operation, the PSNR does not give the ultimate judgment about the quality improvement. However, visual inspection confirms that the proposed method delivers the best results also in terms of perceive stereoscopic quality.

In Figure 5 some visual results are presented, while the full comparison is presented in Figure 7.
<table>
<thead>
<tr>
<th>Method</th>
<th>32</th>
<th>34</th>
<th>36</th>
<th>38</th>
<th>40</th>
<th>42</th>
<th>44</th>
<th>46</th>
<th>48</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bullinger</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocky</td>
<td>36.97</td>
<td>36.06</td>
<td>35.61</td>
<td>34.36</td>
<td>33.72</td>
<td>32.68</td>
<td>31.91</td>
<td>31.38</td>
<td>29.84</td>
<td>29.21</td>
</tr>
<tr>
<td>Proposed method</td>
<td><strong>37.45</strong></td>
<td><strong>36.71</strong></td>
<td><strong>36.38</strong></td>
<td><strong>35.11</strong></td>
<td><strong>34.64</strong></td>
<td><strong>33.33</strong></td>
<td><strong>32.61</strong></td>
<td><strong>32.09</strong></td>
<td><strong>30.45</strong></td>
<td><strong>29.82</strong></td>
</tr>
<tr>
<td>Single channel</td>
<td>37.28</td>
<td>36.52</td>
<td>36.22</td>
<td>34.96</td>
<td>34.44</td>
<td>33.21</td>
<td>32.48</td>
<td>31.98</td>
<td>30.36</td>
<td>29.71</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>37.14</td>
<td>36.37</td>
<td>36.13</td>
<td>34.65</td>
<td>34.11</td>
<td>33.04</td>
<td>32.38</td>
<td>31.80</td>
<td>30.32</td>
<td>29.74</td>
</tr>
<tr>
<td>Bilateral filter</td>
<td>36.91</td>
<td>36.14</td>
<td>35.85</td>
<td>34.40</td>
<td>33.86</td>
<td>32.68</td>
<td>32.22</td>
<td>31.69</td>
<td>30.19</td>
<td>29.48</td>
</tr>
<tr>
<td><strong>Butterfly</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocky</td>
<td>34.95</td>
<td>34.03</td>
<td>32.96</td>
<td>31.77</td>
<td>30.73</td>
<td>29.77</td>
<td>28.78</td>
<td>27.64</td>
<td>26.46</td>
<td>25.38</td>
</tr>
<tr>
<td>Proposed method</td>
<td><strong>35.85</strong></td>
<td><strong>34.98</strong></td>
<td><strong>33.90</strong></td>
<td><strong>32.70</strong></td>
<td><strong>31.61</strong></td>
<td><strong>30.63</strong></td>
<td><strong>29.61</strong></td>
<td><strong>28.41</strong></td>
<td><strong>27.27</strong></td>
<td><strong>26.09</strong></td>
</tr>
<tr>
<td>Single channel</td>
<td>35.41</td>
<td>34.63</td>
<td>33.61</td>
<td>32.43</td>
<td>31.35</td>
<td>30.41</td>
<td>29.43</td>
<td>28.23</td>
<td>27.15</td>
<td>25.93</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>35.17</td>
<td>34.42</td>
<td>33.51</td>
<td>32.07</td>
<td>31.18</td>
<td>30.21</td>
<td>29.28</td>
<td>27.97</td>
<td>27.01</td>
<td>25.98</td>
</tr>
<tr>
<td>Bilateral filter</td>
<td>34.58</td>
<td>33.87</td>
<td>32.91</td>
<td>31.61</td>
<td>30.72</td>
<td>29.75</td>
<td>28.88</td>
<td>27.67</td>
<td>26.71</td>
<td>25.72</td>
</tr>
<tr>
<td><strong>Car</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocky</td>
<td>35.45</td>
<td>34.45</td>
<td>33.40</td>
<td>32.57</td>
<td>31.53</td>
<td>30.48</td>
<td>29.73</td>
<td>28.60</td>
<td>27.95</td>
<td>26.98</td>
</tr>
<tr>
<td>Proposed method</td>
<td><strong>35.99</strong></td>
<td><strong>35.13</strong></td>
<td><strong>34.21</strong></td>
<td><strong>33.40</strong></td>
<td><strong>32.38</strong></td>
<td><strong>31.54</strong></td>
<td><strong>30.65</strong></td>
<td><strong>29.67</strong></td>
<td><strong>28.80</strong></td>
<td><strong>27.84</strong></td>
</tr>
<tr>
<td>Single channel</td>
<td>35.59</td>
<td>34.87</td>
<td>34.00</td>
<td>33.26</td>
<td>32.22</td>
<td>31.37</td>
<td>30.53</td>
<td>29.49</td>
<td>28.66</td>
<td>27.73</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>35.73</td>
<td>34.93</td>
<td>33.95</td>
<td>33.24</td>
<td>32.18</td>
<td>31.32</td>
<td>30.45</td>
<td>29.41</td>
<td>28.67</td>
<td>27.85</td>
</tr>
<tr>
<td>Bilateral filter</td>
<td>35.20</td>
<td>34.42</td>
<td>33.48</td>
<td>32.80</td>
<td>31.76</td>
<td>30.79</td>
<td>30.03</td>
<td>29.03</td>
<td>28.34</td>
<td>27.59</td>
</tr>
<tr>
<td><strong>Horse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocky</td>
<td>32.23</td>
<td>31.08</td>
<td>29.89</td>
<td>28.76</td>
<td>27.78</td>
<td>26.76</td>
<td>25.92</td>
<td>25.09</td>
<td>24.37</td>
<td>23.62</td>
</tr>
<tr>
<td>Proposed method</td>
<td><strong>32.37</strong></td>
<td><strong>31.30</strong></td>
<td><strong>30.11</strong></td>
<td><strong>29.02</strong></td>
<td><strong>28.05</strong></td>
<td><strong>27.13</strong></td>
<td><strong>26.38</strong></td>
<td><strong>25.64</strong></td>
<td><strong>24.90</strong></td>
<td><strong>24.11</strong></td>
</tr>
<tr>
<td>Single channel</td>
<td>32.01</td>
<td>31.02</td>
<td>29.94</td>
<td>28.87</td>
<td>27.92</td>
<td>27.00</td>
<td>26.26</td>
<td>25.53</td>
<td>24.80</td>
<td>24.01</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>31.52</td>
<td>30.66</td>
<td>29.75</td>
<td>28.58</td>
<td>27.68</td>
<td>26.82</td>
<td>25.94</td>
<td>25.41</td>
<td>24.79</td>
<td>24.15</td>
</tr>
<tr>
<td>Bilateral filter</td>
<td>31.52</td>
<td>30.62</td>
<td>29.63</td>
<td>28.46</td>
<td>27.55</td>
<td>26.61</td>
<td>25.60</td>
<td>25.06</td>
<td>24.44</td>
<td>23.84</td>
</tr>
<tr>
<td><strong>Soccer2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocky</td>
<td>34.73</td>
<td>33.58</td>
<td>32.35</td>
<td>31.19</td>
<td>30.28</td>
<td>29.24</td>
<td>28.29</td>
<td>27.39</td>
<td>26.55</td>
<td>25.54</td>
</tr>
<tr>
<td>Proposed method</td>
<td><strong>35.14</strong></td>
<td><strong>34.14</strong></td>
<td><strong>33.03</strong></td>
<td><strong>31.96</strong></td>
<td><strong>31.03</strong></td>
<td><strong>30.02</strong></td>
<td><strong>29.04</strong></td>
<td><strong>28.22</strong></td>
<td><strong>27.23</strong></td>
<td><strong>26.27</strong></td>
</tr>
<tr>
<td>Single channel</td>
<td>34.76</td>
<td>33.86</td>
<td>32.81</td>
<td>31.79</td>
<td>30.85</td>
<td>29.85</td>
<td>28.93</td>
<td>28.09</td>
<td>27.15</td>
<td>26.18</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>34.81</td>
<td>33.87</td>
<td>32.78</td>
<td>31.69</td>
<td>30.75</td>
<td>29.59</td>
<td>28.65</td>
<td>27.85</td>
<td>27.17</td>
<td>26.18</td>
</tr>
<tr>
<td>Bilateral filter</td>
<td>34.99</td>
<td>33.95</td>
<td>32.76</td>
<td>31.67</td>
<td>30.70</td>
<td>29.61</td>
<td>28.72</td>
<td>27.78</td>
<td>27.01</td>
<td>26.04</td>
</tr>
</tbody>
</table>

Table 1. PSNR (dB) results for different QP parameters
2.4 Sharpening of stereo video

2.4.1 Alpha-rooting in 4D-transform domain

As seen in Figure 5, some of the deblocked videos appear to be smoothed and present also some minor loss of contrast and color. It is possible to further improve the quality by image sharpening. The 4D group structure allows doing this in a very elegant way within the transform-domain filtering stage. The transform-domain coefficients are non-linearly transformed by alpha-rooting resulting in a simultaneous sharpening of the left and right images.

The alpha-rooting is applied to the coefficients thresholded by the Wiener threshold in the second stage of the deblocking procedure:

\[
U_{\text{sharp}}(\chi) = \begin{cases} 
\text{sign}(\hat{U}(\chi))\left|\hat{U}(0)\right|^{\frac{1}{h}} & \text{if } \hat{U}(0) \neq 0 \\
\hat{U}(\chi) & \text{otherwise}
\end{cases}
\] (12)

where \( h > 1 \). This value however cannot be chosen too high, as it leads to distorted and unnaturally looking images. It was found experimentally that \( h = 1.09 \) provide a good amount of sharpening with no visible artefacts.
2.4.2 Experimental results

Figure 6 presents the result of deblocking and deblocking combined with sharpening.

Figure 6. Alpha-rooting results; from left to right: blocky image (a), deblocked image (b) and deblocked with sharpening step (c)
As seen in the figure, the resulting images after alpha-rooting exhibit higher contrast and better visual quality than the deblocked ones.

### 2.5 Conclusions

In this work, a new deblocking method for stereo sequences has been presented. Using the redundant information in similar blocks along temporal direction and between the left and right channel, a two step filter is introduced: the results show good quality result and an improvement respect to other deblocking filter, that are over performed in terms of PSNR. Finally a sharpening filter during the last step has been used in order to improve the visual quality of the resulting video.

Fig. represents a visual comparison of all methods compared. In a zig-zag scanning from top to bottom and from left to right, the images are as follows: original frame, blocky version, deblocked by a bilateral filter [13], deblocked by the sliding window DCT algorithm [12], deblocked by the video BM3D algorithm [10] and finally the proposed one.
Figure 7. Visual comparison of deblocking and sharpening methods: for each video, in a zigzag scanning from

top to bottom and from left to right, the images are as follows: original frame, blocky version, deblocked by a

bilateral filter [13], deblocked by the sliding window DCT algorithm [12], deblocked by the video BM3D

algorithm [10] and finally the proposed on.
3 Disocclusion filling

3.1 Introduction

The 'view plus depth' (V+D) format [14], replaces the left and right channels of a stereo video by a single video sequence augmented by its depth map sequence. Depth map refers to a gray-scale image [15], where each value is proportional to the distance of the corresponding pixel in the video frame from the camera, as shown in Figure 9 (b). Due to the fact that the depth maps are characterized by uniform regions, delineated by sharp objects’ borders, they can be easily and efficiently coded. For this reason this format appears to be suitable for 3D video transmission systems with limited bandwidth as the transmission of stereo content reduces to the transmission of 2D video and its relative depth map that represents a little ‘overload’ of the main video channel.

At the receiver side, the Depth-Image-Based-Rendering (DIBR) technique re-generates the stereo sequence from the color, texture and geometry information in the V+D representation. The use of geometric rules [16] together with the knowledge of the distance of the objects from the camera allow the rendering of a new view of the scene, that is an image representing the same scene recorded from a different point of view as shown in Figure 8.

It is possible to represent this procedure by means of the following formula:

$$X_{i,r} = X + \Delta x_{i,r}$$  \hspace{1cm} (13)
where $\Delta x_{l/r}$ is the horizontal shift equal to:

$$
\Delta x_{l/r} = \begin{cases} 
\frac{t_x f}{2Z} & \text{left view} \\
-\frac{t_x f}{2Z} & \text{right view}
\end{cases}
$$

(14)

where $Z$ is the depth value of the pixel in the intermediate image, $f$ is the focal length of the camera and $X_{l/r}$ is the resulting horizontal coordinate of the pixel in the left/right virtual camera [17].

For improving the quality of depth feeling the so-called 'Zero Parallax Setting' (ZPS), a plane in which there is no disparity, is used [18]. With this method it is possible to simulate a virtual shift of the sensors thus adjusting the depth perception [19]. According to Eq. (13), the formula becomes:

$$
X_{l/r} = X + \Delta x_{l/r} + h
$$

(15)

where the sensors' shift $h$ is:

$$
h = \begin{cases} 
\frac{t_x f}{2Z} & \text{left view} \\
\frac{t_x f}{2Z} & \text{right view}
\end{cases}
$$

(16)

In Eq. (16), $Z_c$ is the 'convergence plane', and usually it is set to the intermediate perceived distance.

Figure 9 shows the rendered frames using Eq. (15). It can be observed that after their construction (warping), the new images present some 'black holes', called disocclusions, which are precisely the regions that become visible after the simulated shifting of the focal point.

The problem of dealing with disocclusion holes has been addressed in several works. Some of them suggest pre-processing of the depth map by using different filters. Since disocclusions are generated by vertical discontinuity in the depth map, smoothing of these discontinuities before the rendering phase reduces the size of the holes and facilitates the filling process. After the rendering process, the disocclusions of size 1-2 pixels can be filled with a local averaging filter.

In [20], a symmetric Gaussian filtering of the depth map has been proposed in order to reduce the size of disoccluded areas. The resulting images contain smaller-size holes for the price of higher blur around edges. In [16], Zhang et al. have tried to avoid blurring artifacts by using an asymmetric Gaussian filter. The resulting images do not contain distortions on vertical edges, while such are still present on the horizontal ones. Park et al. [21] have proposed the use of an edge-dependent filter: it smooths the edges with different coefficients depending on the value.
of the gradient in that point. In particular, the smoothing is stronger where the gradient has a large value in the horizontal direction. This method improves the rendering results, though

![Original image and depth map](image)

**Figure 9.** Example of an original image (a) and its depth map (b), and the corresponding left (c) and right (d) rendered images

...disocclusion artifacts are still present. They become more visible when vertical edges are present close to the disocclusion. The above cited methods are based on modification of the original depth map resulting in a possible *depth loss* effect at the rendering phase. However, their main advantage is in the low computational complexity which allows real time application.

In our preliminary work [22], the algorithm of Park, that has delivered best results in terms of PSNR of the reconstructed images, has been compared to two inpainting methods opportunely modified. In this report, a new version of the exemplar-based inpainting algorithm [23] by Criminisi *et al.* is presented. This modification reduces the computational time and facilitates the real time application. The rest of the paper is organized as follows. In Section 3.2 the proposed approach is motivated and presented. In Section 3.3 both the objective and subjective experiments performed for assessing the performances of the proposed method are described and the collected results are presented. Finally, in Section 3.4, the conclusions are drawn.

### 3.2 Proposed approach

In [23], Criminisi *et al.* presented an exemplar-based inpainting method. It is aimed at recovering an unknown image region, a *hole*, by using the information from the surrounding regions while maintaining high quality of the textures in the corrected regions. The procedure can be summed up in three steps:

1. Computation of a priority map;
2. Selection of disoccluded areas (holes) and collection of similar patches based on block matching;
3. Hole filling.

In this approach, edges inside the disoccluded regions are propagated first, followed by processing the smooth areas.

A priority map $P$ contains, for each pixel of the image, the corresponding priority filling order, which is computed as follows:

$$P = D \cdot C$$

(17)

The term $D$ represents the intensity and direction of the edges surrounding the unknown area. It contains values which are large in the case of a high-contrast edge in the hole direction; $C$ indicates the number of known pixels surrounding the unknown current pixel, and $\cdot$ represents the componentwise multiplication of the two matrices.

After the priority map has been computed, the pixel with the highest priority value belonging to the boundary between known and occluded areas is selected, and a target window of size $w \times w$ is centered on it. The known area is used as a template in the similarity matching process. The best match is found and then used to fill the unknown part of the target by substitution.

In the proposed method the three steps have been modified in order to:

i) adapt the method to the particular type of holes, i.e. disocclusions,
ii) to speed up the block matching step, and
iii) to improve the quality of the results.

As can be noticed in Figure 9 (c) and (d), the disocclusions resulting by DIBR techniques are located on the boundaries of objects positioned at different distances from the camera thus resulting in different depth values. By using the classical scheme for computing the priority map, the pixel with the highest priority value may belong to the foreground. In this case the target window is centered on the foreground and regions belonging to the foreground are used for filling the hole. This leads to perceivable artifacts since the disocclusion belongs to the background. To cope with this problem, the use of a modified priority map is proposed. This modification ensures that the filling process is first performed considering areas belonging to the background and then to the foreground.

The depth map contains information about the depth values of all the pixels but for those in the disoccluded areas whose values are unknown. The complete map is obtained after filling the unknown pixel values by some estimation technique. To this aim a rendering algorithm is applied to the depth map. The resulting views present the same disocclusions as the rendered 3D frame, and consequently make it possible to estimate the depth value of the pixels belonging to the boundaries of the disocclusions. The new depth map, $ID_{r/t}$, can be obtained by computing the complement to the maximum luminance value, $L_{max}$, of $D_{r/t}$. Since the background areas are identified with a higher value than the pixels in the foreground, it is possible to estimate the depth values of the occluded areas by performing a smoothing filtering of $ID_{r/t}$.

In the proposed method, the smoothed views are used as priority maps for generating the best filling order of the pixels in the rendered frame. Figure 10 shows the procedure for the priority map computation.
In the proposed approach, the computational complexity of the block matching algorithm used in [23] is reduced. Specifically, the distance between the target window and the possible patches has been computed using only the $Y$ frame component. Another improvement for reducing the computational time is obtained by introducing two thresholds, $\beta$ and $\alpha$, as follows:

- The distance between patches, denoted by $d$, is computed by using only half of the available pixels: if $d$ is with higher value than $\beta$, the current patch is discarded and the following patch is considered, otherwise if $d$ is smaller than $\beta$, the remaining pixels are used and the distance $D$ is stored and compared to the other distances. The less similar patches are discarded by halving the number of operations.
- After the distance $D$ is computed, for further reducing the computational cost needed to find the most suitable patch, $D$ is compared to the threshold $\alpha$ defining the maximum acceptable difference between the patches. If a value $D$ smaller than $\alpha$ is found, the block matching procedure halts and the current patch is used for filling the disocclusion; otherwise, the next patch is considered.

The matching process is sketched in the flowchart in Figure 11.

The selection of the thresholds $\alpha$ and $\beta$ is critical for the method's performances. To this aim, the block matching based method proposed in [22], has been used to fill a set of 5 3D-videos with variable window's size ($5 \times 5$ pixels, $7 \times 7$ pixels and $9 \times 9$ pixels). The average per-pixel distance between each original and best match patch has been evaluated to analyze the overall error trend. From Figure 12 it can be noticed that the error relative to the $95^{th}$ percentile is below 5. According to the performed test $\beta$ has been set equal to 5 and $\alpha$ equal to 1.
For further reducing the computational complexity, the search region used in the similarity matching procedure, is reduced to a window of size $M \times M$ pixels around the target in the current frame and it is extended to the previous and successive $N$ frames in the temporal domain. In this way, moving objects are considered and the disocclusions replacement can be performed by also using the information revealed by the objects' movements.

A further improvement is achieved by increasing from $1$ to $k$ the number of similar patches exploited for recovering the occluded areas. A weighted non-local mean of the $k$ patches is performed for estimating the filling part, as shown in Eq. (18) and (19). The coefficients are proportional to the distance between target and patches:

$$\psi'_t = \sum_{i=1}^{k} \frac{w(\psi'_t) \psi_i}{\sum_{i=1}^{k} w(\psi'_t)}$$  \hspace{1cm} (18)

and

$$w(\psi'_t) = e^{-\frac{d(\psi'_t, \psi_i)}{h}}$$  \hspace{1cm} (19)

where $\psi'_t$ is the region of the target window to be recovered, $\psi_i$ are the most similar $k$ patches and $w(\psi'_t)$ are their relative weights.

In order to adapt the algorithm to the use of the weighted mean of more patches, it is necessary to modify the threshold $\beta$ by taking into account the fact that the error relative to the 95th percentile increases; in the performed experiments, in which $k$ has been selected equal to 5, $\beta$ has been chosen equal to 7.
3.3 Experimental results

To verify the effectiveness of the proposed techniques, experimental tests have been performed. As described in the previous Section, the raw impaired videos have been processed with different algorithms and the quality of the resulting data assessed by means of both objective and subjective experiments.

Twenty sequences were drawn from a pool of uncompressed stereo videos: Horse, Car, Butterfly, Bullinger and Quest [24].

The video content has been chosen to provide a range of different situations in motion, detail, color, contrast, brightness, and depth perception, as described below:

- Horse: outdoor scene with slow motion and two main depth levels;
- Car: outdoor moving scene with constant moderate motion;
- Butterfly: cartoon video with fixed camera, different depth levels and moving objects;
- Bullinger: news type of scene with a speaker in foreground and still background (two main depth levels);
- Quest: cartoon video with varying depth levels.

The original videos are in the format of two-channel stereo, which have been used to estimate the depth maps aligned with the left channel [24]. In order to mitigate possible flickering effects in rendered videos, all estimated depth maps have been further processed with a modified bilateral Gaussian filter presented in [25].

The test set is composed by 5 original 3D videos and 15 processed sequences. The processed sequences are obtained by rendering the right channel using the original left channel and the estimated associated depth followed by processing the reconstructed sequences with three occlusion-handling algorithms denoted as: Oliveira [26], Park [21], and the proposed one.

All video sequences are 10 s long and the size of each frame is 427 × 240 pixels. The frame rate for the sequence Bullinger is 30 fps, while for the other videos is 15 fps. Sample frames that have been extracted from the original left sequences are shown in Figure 13.
3.3.1 Objective tests

In this work, the objective quality of the rendered sequences has been evaluated by using state-of-the-art quality assessment methods. Given a luminance frame $I(x,y)$ and its impaired version $I'(x,y)$, the objective quality can be rated by using:

- Peak Signal to Noise Ratio (PSNR):

$$PSNR(dB) = 10 \log_{10} \left( \frac{\text{max}(I(x,y)^2)}{MSE} \right)$$

(20)

where $MSE = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} [I(x,y) - I'(x,y)]^2$.

- $\text{PSNR}_{\text{HVS}}$ [27] that is a modified version of PSNR that takes into account both the Contrast Sensitivity Function (CSF) and the between-coefficient contrast masking in the
Discrete Cosine transform (DCT) domain. This metric has been demonstrated to rank impaired images closely to the human judgment [27],

- Weighted Peak Signal to Noise Ratio (WPSNR) [28], a pixelwise comparison metric as the PSNR, whose ranking is modified according to the amount of texture in the image:

\[
WPSNR(dB) = 10\log_{10}\left(\frac{\max(I(x, y)^2)}{\|NVF(I'(x, y) - I(x, y))\|^2}\right)
\]

(21)

where NVF is the Noise Visibility Function whose value is 1 in flat regions and 0 in textured regions and edges [28];

- Video Quality Metric (NTIA-VQM) [29] is a Full Reference standardized system for quantifying the perceptual quality degradation in video systems using compression. The scores are reported on a nominal range of [0, 1], where zero indicates excellent quality. Based on good correlation with subjective human rating, it has been adopted as standard and international recommendation [30], [31].

- NRMos [32] is a No Reference metric for assessing the quality of impaired videos. It is based on the analysis of the inter-frame correlation measured at the output of the rendering application. It does not require information on the errors, delays, and latencies affecting the link and on the countermeasures introduced by decoders in order to face the potential quality loss. The quality is assessed with a score in the range [0, 5], where 5 stands for excellent quality.

The obtained results have been averaged on the video database and are reported in Table 2. All adopted quality metrics show an improvement in the processed video frames, with respect to the raw ones (raw denotes rendered channels with no holes filling). The Criminisi based method, slightly outperforms the other ones.

Table 3 shows the values obtained by the quality evaluation of each sequence in the database.

<table>
<thead>
<tr>
<th>Algorithm/Metric</th>
<th>PSNR</th>
<th>PSNR&lt;sub&gt;HVS_M&lt;/sub&gt;</th>
<th>WPSNR</th>
<th>NTIA-VQM</th>
<th>NRMos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>27.78</td>
<td>25.52</td>
<td>36.26</td>
<td>0.31</td>
<td>4.90</td>
</tr>
<tr>
<td>Oliveira based</td>
<td>26.78</td>
<td>24.14</td>
<td>34.74</td>
<td>0.38</td>
<td>4.84</td>
</tr>
<tr>
<td>Park based</td>
<td>27.56</td>
<td>25.15</td>
<td>35.99</td>
<td>0.31</td>
<td>4.85</td>
</tr>
<tr>
<td>Raw</td>
<td>23.47</td>
<td>20.88</td>
<td>31.58</td>
<td>0.53</td>
<td>4.42</td>
</tr>
</tbody>
</table>

Table 2. Average objective results computed by using state of the art metrics

---

1 PSNR<sub>HVS_M</sub> Matlab code is freely available for download at http://www.ponomarenko.info/psnrhsvsm.htm
<table>
<thead>
<tr>
<th></th>
<th>PSNR</th>
<th>PSNR_HVS</th>
<th>WPSNR</th>
<th>NTIA-VQM</th>
<th>NRMs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bullinger</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>26.35</td>
<td>24.21</td>
<td>34.84</td>
<td>0.33</td>
<td>5.00</td>
</tr>
<tr>
<td>Oliveira based</td>
<td>26.33</td>
<td>23.53</td>
<td>34.00</td>
<td>0.37</td>
<td>4.90</td>
</tr>
<tr>
<td>Park based</td>
<td>26.30</td>
<td>23.80</td>
<td>34.69</td>
<td>0.34</td>
<td>5.00</td>
</tr>
<tr>
<td>Raw</td>
<td>23.74</td>
<td>20.84</td>
<td>31.04</td>
<td>0.57</td>
<td>4.56</td>
</tr>
<tr>
<td><strong>Butterfly</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>27.84</td>
<td>24.55</td>
<td>36.14</td>
<td>0.35</td>
<td>4.80</td>
</tr>
<tr>
<td>Oliveira based</td>
<td>25.68</td>
<td>21.34</td>
<td>32.81</td>
<td>0.42</td>
<td>4.94</td>
</tr>
<tr>
<td>Park based</td>
<td>27.53</td>
<td>23.94</td>
<td>35.56</td>
<td>0.36</td>
<td>4.83</td>
</tr>
<tr>
<td>Raw</td>
<td>16.6</td>
<td>12</td>
<td>23</td>
<td>0.89</td>
<td>3.59</td>
</tr>
<tr>
<td><strong>Car</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>29.43</td>
<td>27.75</td>
<td>37.12</td>
<td>0.30</td>
<td>4.94</td>
</tr>
<tr>
<td>Oliveira based</td>
<td>29.48</td>
<td>27.86</td>
<td>37.18</td>
<td>0.30</td>
<td>4.96</td>
</tr>
<tr>
<td>Park based</td>
<td>29.24</td>
<td>27.51</td>
<td>36.88</td>
<td>0.30</td>
<td>4.96</td>
</tr>
<tr>
<td>Raw</td>
<td>25.88</td>
<td>24.83</td>
<td>34.29</td>
<td>0.47</td>
<td>4.66</td>
</tr>
<tr>
<td><strong>Horse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>25.92</td>
<td>25.10</td>
<td>36.11</td>
<td>0.24</td>
<td>4.89</td>
</tr>
<tr>
<td>Oliveira based</td>
<td>25.98</td>
<td>25.08</td>
<td>36.05</td>
<td>0.24</td>
<td>4.90</td>
</tr>
<tr>
<td>Park based</td>
<td>25.44</td>
<td>24.75</td>
<td>35.89</td>
<td>0.24</td>
<td>4.66</td>
</tr>
<tr>
<td>Raw</td>
<td>22.09</td>
<td>20.27</td>
<td>31.79</td>
<td>0.49</td>
<td>4.37</td>
</tr>
<tr>
<td><strong>Quest</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>29.37</td>
<td>25.99</td>
<td>37.07</td>
<td>0.32</td>
<td>4.86</td>
</tr>
<tr>
<td>Oliveira based</td>
<td>29.00</td>
<td>25.57</td>
<td>36.63</td>
<td>0.39</td>
<td>4.86</td>
</tr>
<tr>
<td>Park based</td>
<td>29.27</td>
<td>25.76</td>
<td>36.92</td>
<td>0.34</td>
<td>4.83</td>
</tr>
<tr>
<td>Raw</td>
<td>26.46</td>
<td>23.75</td>
<td>34.78</td>
<td>0.44</td>
<td>4.60</td>
</tr>
</tbody>
</table>

Table 3. Objective results computed by using state of the art metrics for the sequences: Bullinger, Butterfly, Car, Horse, and Quest
3.3.2 Subjective tests

The methods for disocclusion filing have been compared also against quality judgments of persons in subjective tests.

Video clips were presented to the viewers on a portable 3.1” autostereoscopic display with horizontally double-density pixel arrangement created by NEC Technologies [33].

The viewers ranked the test sequences according the Single Stimulus Continuous Quality Evaluation SSCQE [34] approach.

In total, 33 subjects with age between 22 and 40 participated to the tests.

The test has been divided into 3 sections, namely:

- **Preliminary vision test**: this is used to check for possible vision disabilities of the test persons, e.g. binocular blindness;
- **Training test**: this is used to train the observers to rank the videos for their quality on the given scale;
- **Quality test**: this is the core of the test. In this phase each user is asked, after each video, to rank the quality of each sequence, as follows:
  - **Acceptance of the video**: each observer has to say if the quality of the video is acceptable (yes/no);
  - **Overall video quality**: each observer has to judge the overall video quality by expressing his/her opinion with a number in the range \([0, 10]\) where 0 corresponds to the lowest quality and 10 to the highest;
  - **Perceived depth**: each subject evaluates the perceived sensation of depth with a number in the range \([0, 10]\) where 0 corresponds to the lowest level of perceived quality and 10 to the highest.

The overall test per subject has been about 40 minutes long.

First of all the acceptance of the videos has been evaluated. The collected results are expressed in percentage and are shown in Figure 14. All sequences but *Quest* have reached acceptance rates above 50%. As of the *Quest* sequence, the accuracy of the estimated depth around foreground objects’ borders has been rather low. It could not be corrected with the post-filtering technique applied. The low-quality depth caused artifacts in the rendered foreground objects and various hole-filling algorithms made no difference for the evaluating subjects who rejected the corresponding videos. Note that the results of the objective comparisons did not detect that problem, as the rendering artifacts are in relatively small areas and will negligible effect on the overall score. Due to the low acceptance rate, the quality results of the *Quest* sequence are excluded from the analysis thereafter.

The acceptance rate averaged over the rest of four sequences is depicted in Figure 17 (c). The figure shows quite good acceptance of all of the methods compared.
Figure 14. Acceptance evaluations for Bullinger (a), Butterfly (b), Car (c), Horse (d) and Quest (e) sequences
The quality of the rendered sequences and the perceived depth have been evaluated by means of the Mean Opinion Score (MOS). The obtained results and the confidence intervals at 95% for individual sequences are shown in Figure 15 and Figure 16.

The quality and the perceived depth score averaged over four sequences are shown in Figure 17 (a) and (b).

As seen in Figure 15, the proposed method performs in the most consistent way along different sequences. It is the best for three of the sequences and equally good to the Park’s technique for the sequence *Butterfly*, which represents a synthetic content. While there is similar trend for the different contents, there are higher scores for the sequences *Butterfly* and *Horse*. In qualitative description after the tests, most of the subjects mentioned that those contents were more appealing in terms of likable foreground objects and vivid colors. The overall score calculated over four contents, as shown in Figure Figure 17 (a), also confirms the superiority of the proposed method. As far as the perception of depth is concerned, Figure 16, there is higher uncertainty among techniques and contents manifested by the wider confidence intervals. Comparing Figure 15 with Figure 16, one could conclude that the disocclusion effects are mostly perceived as 2D type of artifacts influencing the overall quality, as exemplified by Figure 17 (a), while their effect to the depth perception is rather marginal (Figure 17 (b)).

![Figure 15. Quality MOS for Bullinger (a), Butterfly (b), Car (c), and Horse (d) sequences](image-url)
Figure 16. Perceived depth MOS for Bullinger (a), Butterfly (b), Car (c), and Horse (d) sequences.
3.4 Conclusions

In this section, a modified version of the inpainting technique has been presented, aiming at efficient and high-quality disocclusion handling in DBIR. To this purpose, a weighted mean of the \textit{k} most similar patches has been used for recovering the disoccluded regions in a more efficient way. Improvements have been suggested for speeding up the search for similar patches and improving the quality by proper prioritization and non-local mean weighting. In order to prove the feasibility of the method, objective and subjective tests have been performed, comparing the technique with the technique by Park [21] and the modified version of the inpainting technique by Oliveira [22]. The developed technique shows high and consisting ranks over various video contents with reduced computational cost.
4 Error Concealment

With the increasing number of 3D video applications, 3D error concealment techniques have become a very important field of interest. Since in stereoscopic video sequences, not only temporal and spatial correlations of mono videos, but inter-view correlations exist, new methods are proposed. In depth-image based rendering, where a colour video and a depth video sequences are used, deep video and colour video are highly correlated. Using this fact, there are several methods to conceal entire frame losses. In [35], Hewage et al. use shared motion vector for depth and colour video in encoding phase and send those motion vectors redundantly. When a packet loss occurs, for instance colour video frame’s motion vectors are lost, corresponding depth motion vectors, which are the same as the lost motion vectors, are used and the lost frame is concealed. This method is very efficient and the only error is caused because of the lost residual data. However using shared motion vectors in the encoding phase increases the bandwidth of the residual data. In [36], Liu et al. also use colour video motion vectors for the depth frame losses. They also extract motion edges using depth video motion vectors for the colour frame losses. This method is more efficient than [35] in terms of bitrate, and also no additional step in encoding phase is needed. However, the motion edge extraction and the concealment phases are computationally complex.

In this work, we propose two computationally less complex methods for error concealment of 2D+depth video. The first one is to conceal the lost frame using the last frame and the current complementary frame, and render. The second is to use the complementary motion vectors for the lost frame. We also show the effect of complementary motion vectors for MVC coding with simplified structure.

4.1 Concealment of V+D video

4.1.1 Concealment Using Complementary Frame

For this method, one frame is considered to be lost each time, and using the last frame and current complementary (if depth frame is lost, complementary is color (left) video, and if color (left) frame is lost, complementary is depth video) frame, right frame is rendered. For instance at time (t+1), the left frame is lost as in Figure1. The left frame at time (t) and the depth frame at time (t+1) are used and right frame at time (t+1) is rendered. If the depth frame is lost at time (t+1), the same process is used for left frame at time (t+1) and depth frame at time (t).
Figure 18: Rendering from left and depth frames

The following graphs are obtained by repeating the explained method for each frame. In the graphs, “rendered” means the concealment is done with the proposed method, and the “last” means the concealment is done with simply by frame copy. The SSIM and PSNR are calculated using concealed right frames where the reference frame is the original right video.

Figure 19: PSNR and SSIM results when left frames of “Car Movie” are lost

Figure 20: PSNR and SSIM results when left frames of “Hands Movie” are lost
Figure 21: PSNR and SSIM results when left frames of “Rollerblade Movie” are lost

Figure 22: PSNR and SSIM results when left frames of “Heidelberg Alleys Movie” are lost

Figure 23: PSNR and SSIM results when left frames of “Horse Movie” are lost
Figure 24: PSNR and SSIM results when left frames of “Knights Quest Movie” are lost

Figure 25: PSNR and SSIM results when left frames of “Rhine Valley Movie” are lost

Figure 26: PSNR and SSIM results when depth frames of “Car Movie” are lost
Figure 27: PSNR and SSIM results when depth frames of “Hands Movie” are lost

Figure 28: PSNR and SSIM results when depth frames of “Heidelberg Alleys Movie” are lost

Figure 29: PSNR and SSIM results when depth frames of “Horse Movie” are lost
Figure 30: PSNR and SSIM results when depth frames of “Knights Quest Movie” are lost

Figure 31: PSNR and SSIM results when depth frames of “Rhine Valley Movie” are lost

Figure 32: PSNR and SSIM results when depth frames of “Rollerblade Movie” are lost
4.1.2 Concealment Using Complementary Motion Vectors

For this method, H.264 encoder and decoder is used (ref). In case of frame loss, motion vectors are used in complementary frame decoding process. For instance when a left movie is decoded, and Kth frame is considered to be lost, then the Kth frame’s motion vectors of the depth video is used and the left frame is decoded with those motion vectors and no residual data (Figure 33). In all the experiments, motion vectors of 4x4 MBs are used (Figure 34-45).

![Diagram](Image)

Figure 33: Rendering using complementary motion vectors (mD(t+1) corresponds to depth motion vectors)
Figure 34: PSNR performance comparison for ‘concealment with last frame’ and ‘concealment with 16x16 depth MV copy method’ (Heidelberg Movie)

Figure 35: SSIM performance comparison for ‘concealment with last frame’ and ‘concealment with 16x16 depth MV copy method’ (Heidelberg Movie)
Figure 36: PSNR performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Heidelberg Movie Left View)

Figure 37: SSIM performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Heidelberg Movie Left View)
Figure 38: PSNR performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 left MV copy method’ (Heidelberg Movie Depth View)

Figure 39: SSIM performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 left MV copy method’ (Heidelberg Movie Depth View)
Figure 40: PSNR performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Knights Quest Movie Left View)

Figure 41: SSIM performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Knights Quest Movie Left View)
Figure 42: PSNR performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Rhine Valley Movie Left View)

Figure 43: SSIM performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Rhine Valley Movie Left View)
Figure 44: PSNR performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Rollerblade Movie Left View)

Figure 45: SSIM performance comparison for ‘concealment with last frame’ and ‘concealment with 4x4 depth MV copy method’ (Rollerblade Movie Left View)
4.1.3 Evaluation of the results

"Concealment using Complementary Frame" method does not give good results for left frame losses. As can be seen from Figure 19 to Figure 25, the SSIM and PSNR results for concealed frame is nearly the same as directly copying last frame. Since rendering process has high computational cost, concealing by frame copy is preferable for left frame losses instead of "Concealment using Complementary Frame" method. However for the depth frame losses, implemented method yields better performance than copying last frame. From Figure 26 to Figure 32, it can be seen that concealment with rendering improves PSNR by 1 dB on average. For some scenes, the PSNR of the method is 5 dB higher from the PSNR of copying last frame, where in those scenes depth image has clear entities and edges. In other words, when a scene contains objects which are not close to each other in z axis and moving in x or y axis, concealing with rendering technique performs best results. Even though, the proposed method gives better results, in the figures, it can be noticed that for some cases simply copying the last frame gives better results (up to 8 dB). To explain those, by examining the depth images, it is noted that even for a still scene, sometimes depth image brightens and darkens. For those cases copying last frame is preferable. Another case might be the camera turn around right recorder. In this case the left view changes rapidly although right view remains almost still. For this case also, copying last right view gives better results.

In the second method, "Concealment Using Complementary Motion Vectors", as can be seen from Figure 34 to Figure 45, especially for left frame losses the implemented method yield better results than both copying last frame and the former method. PSNR is improved up to 8 dB. This method is better even in the cases where copying last frame gives better results. The difference is not greater than 1 dB and mostly less than 0.2 dB when frame copy is better. This means that by not checking the profile of the scene, for instance whether it is a still scene, or there are camera panning and so on, this method can be used. Another good aspect of the method is, it does not require higher performance like rendering. The only need is some extra memory. To save the left and depth motion vectors, frame size/16 bytes of memory (because each motion vector is saved for a 4x4 block) is needed. The remaining part is the same decoding process. The only performance degradation can occur at skip frames, where instead of decoding with motion vectors simply copying the last frame is required. Also using this method, there is no need to make changes at encoding part as in [35].

Throughout the simulations on different video sequences, it is noted that although 16x16 MV concealment has less computational cost, it does not improve SSIM and PSNR results. This is a result of not modifying encoding part. The complementary video, where complementary MV’s are taken from, are decoded such that, intra and inter coding techniques are allowed. Within a P coded frame, blocks might be coded 16x16, 8x16, 16x8, 8x8 and so on. Using only 16x16 concealment is not proper at the smaller complementary coded block areas. So the best result can be obtained by making smaller block decoding. And since the smallest block at the decoding process has 4 by 4 size, using 4x4 MV concealment gives best results.

Another point that can be noted throughout simulating different video sequences having different profiles is, for skip or mostly still frames, frame copy gives very similar results with the second method. The only problem is as mentioned before, at still scenes, sometimes depth frames brightens or darkens. For those cases, frame copy gives better results. Also it can be noted that, throughout the simulations, the second method gives better results where the depth image has strong edges, meaning that the scene contains closer and farther objects. The PSNR
performance increases when the objects in the scene moves parallel to the x-y plane. The scenes where an object moves towards or away from the camera, since in the original bitstream the object is mostly encoded with intra blocks, the PSNR performance degrades.

An improvement on the results can be obtained by concealing frames where the complementary frames are encoded in 4x4 intra modes. This ensures having a strong MV data for each lost MB in the concealed frame.

4.2 Concealment for MVC

4.2.1 Concealment Using Complementary Motion Vectors for MVC

The second method, replacing with complementary motion vectors, is also used for stereo videos. For the lost frames, the decoding process is continued using other view’s motion vectors. The process is explained in Figure 46 and the results are shown in Figure 47-63.

Figure 46: Concealment by using complementary motion vectors
Figure 47: PSNR performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Heidelberg Movie Left View)

Figure 48: SSIM performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Heidelberg Movie Left View)
Figure 49: PSNR performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Heidelberg Movie Right View)

Figure 50: SSIM performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Heidelberg Movie Right View)
Figure 51: PSNR performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Knights Quest Movie Left View)

Figure 52: SSIM performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Knights Quest Movie Left View)
Figure 53: PSNR performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Knights Quest Movie Right View)

Figure 54: SSIM performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Knights Quest Movie Right View)
Figure 55: PSNR performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Rhine Valley Movie Left View)

Figure 56: SSIM performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Rhine Valley Movie Left View)
Figure 57: PSNR performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Rhine Valley Movie Right View)

Figure 58: SSIM performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Rhine Valley Movie Right View)
Figure 59: PSNR performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Rollerblade Movie Left View)

Figure 60: SSIM performance comparison for concealment with last frame and concealment with 4x4 “right MV” copy method (Rollerblade Movie Left View)
Figure 61: PSNR performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Rollerblade Movie Right View)

Figure 62: SSIM performance comparison for concealment with last frame and concealment with 4x4 “left MV” copy method (Rollerblade Movie Right View)
4.2.2 Evaluation of the results on stereo concealment

Similar to the utilization of depth motion vectors, use of complementary motion vectors gives better performance results for camera movements. Especially when the moving region in the video is large, or far from the camera, use of the complementary motion vectors gives better results, because the projection of the motion vectors of the complementary view is relatively less affected by the viewing angle.
5 Implementation of video processing algorithms

The modern video enhancement algorithms are considerably complex, and therefore the limitations of mobile devices have to be taken into consideration when balancing the needed computational power and the quality requirements. Bilateral filter, sliding window DCT and VBM3D are algorithms that have been considered in the Mobile 3DTV project both for denoising and deblocking of stereo video. As the simplest of these three, bilateral filter was chosen as a main test subject. The purpose was to explore, what kind of possibilities for implementation is offered by a platform corresponding to the level of current devices on the market. A similar implementation of sliding windowed DCT is also described. Based on the similarities of SW-DCT and VBM3D, estimates on its computational requirements are also made.

Ideally the implementation should be able to process stereo video in real time while it is being watched, i.e. in the range of 25-30 frames per second. Because of the processing power of the OMAP 3 platform chosen as a test bench is somewhat limited, the possibility of using OpenCL in future mobile products is also considered. A GPU of a netbook, considered as a reasonable substitute of currently missing GPUs supporting OpenCL on mobile platforms, has been used for performance testing of the OpenCL implementation.

5.1 Algorithms

5.1.1 Bilateral filtering

Bilateral filter takes into consideration not only the spatial domain, but also the color information (range) of the neighborhood when calculating the weights for any given pixel. This gives it the ability to preserve edges, which would be smoothed out by a traditional spatial filter such as a Gaussian [37].

The bilateral filter needs two functions to map the differences in value to weight values, the spatial coefficients $\psi(x)$ and range coefficients $\phi(x, y)$. Those functions can be arbitrary, but a suitable candidate is for instance the Gaussian. For a rectangular 2-dimensional window $W$ of size $2r + 1 \times 2r + 1$ around image point $x$, the discreet response of the bilateral filter can be expressed as

$$y(x) = \frac{1}{\kappa} \sum_{k \in W} \psi(k) l(x + k) \phi(l(x) - l(x + k)),$$  \hspace{1cm} (22)

where $\kappa = \sum_{k \in W} \psi(k) \phi(l(x) - l(x + k))$ is a normalization term. The normalization term has to be computed for each point the filter is applied to, unlike in a purely spatial filter [38].

5.1.1.1 Constant spatial filter

Even though the filter is eventually applied like a regular convolution, the varying weights of the filter make it considerably heavy to compute. Three ways to speed up the processing were introduced by Porikli [38]. The simplest and fastest of these is that by setting the spatial filter coefficients $\psi(k)$ to a constant value $c$ (i.e. a box filter), the response of the filter can be modified into

$$y_b(x) = c \frac{1}{\kappa} \sum_i h_x(i) \phi(l(x) - i),$$  \hspace{1cm} (23)

where $h_x(i)$ is the histogram of the windowed image around the point $x$. As all spatial locations have the same weight, the spatial location of the range values no longer matters. Therefore it is sufficient to form the response by accumulating over the bins of the histogram, not over single
pixels. This approach has the added benefit of not being sensitive to the size of the window in terms of performance, assuming the computation of the histograms is not either. However, the performance is $O(n)$ in terms of the amount of histogram bins used.

5.1.1.2 Distributive histograms
Porikli originally suggested integral histograms as the tool for computing the histograms needed for Eq. (23) [38]. Another method for this is the distributive histogram proposed by Sizintsev et al., which has a lower computational cost and memory requirement. The distributive histogram is based on the idea that the histograms of two consecutive sliding windows differ only by the histograms of the $1 \times (2r + 1)$ column segments that are included in one but not the other. If assuming the sliding window is traveling along the rows, the new local histogram can be computed from the previous by adding the histogram of the new column segment and subtracting the histogram of the column segment leaving the window. Similarly, computing the histogram of a column segment is done by respectively subtracting and adding the value of the two single pixels that leave and enter the window from the histogram of the segment above it [39].

5.1.2 Sliding Window DCT
Sliding window DCT is a straightforward image enhancement method exploiting sparse representations of overlapping windowed sections of image areas.

![Block diagram of the sliding window DCT](image)

Figure 63 Block diagram of the sliding window DCT. Overlapping blocks are DCT thresholded separately, then combined as a weighted average using the accumulation and weight buffers to create the enhanced image as the output of the filter. Diagram adopted from [40].

The output of the filter at each pixel location is constructed from the over-complete set of windows that included the pixel.

5.1.3 VBM3D
VBM3D is in a sense an extension of the sliding window DCT. Doing a thresholding on the 2D DCT coefficients of each block is replaced with a more complex operation. The windows (i.e. blocks) are matched to others in order to find similar areas either in the same frame or in the temporal direction. This adds a non-local fashion to otherwise locally adaptive sliding DCT processing. The matching blocks are then stacked into a 3-dimensional block. Collaborative filtering is then applied to the 3D block, i.e. DCT is applied in each dimension, and the resulting
coefficients are thresholded. The blocks are then inverse transformed back into spatial domain, and the 2D blocks are returned to their original positions [41]. The construction of the final image follows as in sliding window DCT.

5.2 Implementation environment

5.2.1 OMAP

OMAP is an integrated hardware platform for mobile devices, offering the essential parts for a lightweight mobile computer [42]. OMAP version 3 is used for instance in the Nokia N900 smartphone [43] and Archos tablet devices [44]. The particular device used as the OMAP device in these implementation experiments is a development kit designed by Mistral Solutions, equipped with an OMAP 3530.

The OMAP3 platform consists of several subsystems. For instance, in a 3530 series processor there are three main cores – ARM Cortex A-8 (CPU, or in TI's terminology, a GPP, General Purpose Processor), TMS320 C64x+ (DSP, Digital Signal Processor) and POWERVR SGX Graphics engine (GPU). Also included are several peripherals for functionality such as USB, memory cards and serial port access [45].

The ARM processor runs at around 600-700 MHz depending on settings and is responsible for running the operating system and other regular programs. It comes with a NEON SIMD (Single Instruction Multiple Data) coprocessor, which can be utilized to speed up vectorizable operations, such as convolution used commonly in image processing.[46] The GPU is responsible for rendering graphics to the display outputs on the screen. It supports such standards as OpenGL ES and OpenGL VG.

5.2.1.1 Digital Signal Processor

The TMS320 C64x+ in the OMAP 3530 is a fixed point DSP running at 520MHz. Similarly to the NEON functionality on the ARM processor, it is optimized for signal processing related tasks such as multiply-and-accumulate, while lacking the flexibility of a more generic CPU. There are choices in methods which can be used to interact between the ARM and DSP cores. Both cores have access to the random access memory included on the OMAP. The memory can be mapped according to one’s needs into segments either dedicated to the buffers and program code of the ARM or the DSP, or as shared memory to pass large amounts of data between the two. Operation between them can be coordinated by the use of a messaging protocol. This method of communicating with the DSP corresponds to the one used by the sample application “readwrite”, which is provided in the TI's SDK [47].

In certain conditions, the DSP is capable of performing 8 simultaneous instructions. This is dependent on the structure of the code - if consecutive iterations of a loop are simple enough and do not access the same memory locations, the loops can be unrolled, i.e. performed in parallel. A critical part of implementation is planning the program running on the DSP so that this can be utilized as much as possible. The compiler detects and exploits these kinds of possibilities either automatically or according to the information given to it by the programmer using preprocessor directives and variable qualifiers in the source code. High level control over the compiler's optimization is achieved by the use of compiler flags [48].

5.2.2 OpenCL

GPGPU (General-Purpose computation on Graphics Processing Units) exploits the massive parallel processing capabilities of modern graphics processing units, enabling their use in various applications requiring heavy computation. OpenCL (Open Computing Language) is an open standard for parallel computing, maintained by the Khronos Group. It is designed for controlling heterogeneous computing devices, namely GPUs and multi-core CPUs. The standard
gives device manufacturers a common interface that their products should implement, either in hardware or software. This enables parallel computing software to be written without having to consider the underlying, hardware dependent details.

OpenCL is also making its way towards mobile platforms, with at least Texas Instrument’s newest addition to the product line, OMAP 5, listing official OpenCL 1.1 support [49], and some of the current mobile GPUs having some level of OpenCL compliance [50]. OpenCL is a promising technology for mobile devices due to GPU’s significantly lower energy consumption in comparison to generic CPUs while performing similar tasks. For instance, Nokia’s experiments with image processing algorithms showed the GPU in OMAP making the same operations at only 7% of the energy consumption as the ARM CPU [50].

As there are no properly supported OpenCL devices for mobile usage commonly available yet, the device used to conduct the experiments was an Nvidia ION GPU. The ION is lightweight graphics card meant for relative small devices, such as the Asus 1201PN netbook that was used here.

In OpenCL terminology, the hardware containing stream processors (cores) is called a compute device. A single stream processor sequentially executes a kernel on a number of work items (threads), and the work items can be dynamically organized by the programmer into workgroups for synchronization and memory sharing purposes. The memory used by the application is split into four segments, ranging from host memory used by the CPU, to private memory available to a single work item. Segments of memory differ in their performance and availability. For instance, host memory can be accessed by all work items, but is generally quite slow. Private memory on the other hand can only be seen by the work item owning it, but is fast to use.

The standard defines an API (Application Programming Interface), which is essentially a set of C functions, which are called by an application requiring access to the compute devices. Among other things, the functions allow for querying device information, creating buffer objects and assigning and reading data to and from compute devices.

The other part of the standard is an instruction set, OpenCL C, which is used for programming a single stream processor by writing a kernel. When the application is executed, the kernel is uploaded to the stream processors of the selected compute devices. OpenCL C is a modified version of ISO C99, offering support for vectorized datatypes and synchronization, but lacking some C functionality like recursion, function pointers and the standard C99 headers.

When designing an OpenCL application for a GPU with performance in mind, there are certain aspects that should be taken into consideration, which differ from regular CPU programs. Computation is fast, but the overhead of some operations will be a significant performance loss if not accounted for. For instance, moving data between devices, so data accesses, especially to the host memory, should be minimized. Also the program flow should be similar between threads, as the GPU is optimized to perform a lot of similar operations. If different threads branch out to perform different tasks, this gain is lost. [52]

5.3 Implementation

5.3.1 Bilateral filter on OMAP

To begin with, a simple C++ version with a Matlab MEX interface of the basic algorithm described in chapter 5.1.1 was available as reported in [55]. The first experiments with a bilateral filter were converting this to pure C to run it on the ARM processor, and then replacing the mathematical operations with fixed point arithmetic using the IQMath library [53] to be able to run it on the DSP. The pure ARM version took approximately 40 seconds per frame and when enabling the NEON support for simultaneous operations, dropped down to 20 seconds per frame. The DSP conversion without any DSP specific optimization was showing extremely poor
performance with up to 90 seconds per frame. Due to these very unimpressive preliminary results, attention was directed to a more performance oriented option, the constant spatial filter. Even though it would have been possible to significantly improve that performance, it was unlikely that acceptable levels could have been reached, considering the objective of real time video processing.

The constant spatial filter implemented on the DSP performed at 16 seconds per frame with 64 bin histograms. By modifying the code to allow the compiler to unroll some of the loops, most importantly the summation described in Eq.(23), the performance increased to 6 seconds per frame. Any further optimization efforts were for the most part unsuccessful, only providing insignificant improvements.

The earlier attempts had only used the ARM core to initialize the program and for file I/O. While the DSP was working on the filtering, the ARM was doing nothing. When building distributive histograms was moved to the ARM, the simplification of DSP side control structures gave a larger performance boost than could have been expected from moving the less computationally expensive histogram operations to be done in parallel. Some compromises regarding the accuracy on computation had to be made to further increase performance. The use of the fixed point arithmetic library was removed altogether, and all DSP side math was converted into half-word (16 bits) integer operations with simulated decimal accuracy up to 3 decimals. This in addition with reducing the amount of histogram bins from 64 to 16 brought the time down to 0.66 seconds per frame.

5.3.2 Bilateral and hypothesis filter in OpenCL

An alternative approach has considered implementing the bilateral filter on a GPU. Even though not yet commonly available on the mobile market, OpenCL can be expected to be a viable method of exploiting the hardware of next generation mobile devices. This will give an idea how well the algorithm deploys to the platform.

The distributive histogram method used on the OMAP implementation is an iterative process. It would be very poorly suited on a heavily parallelized platform such as the GPU. Therefore the algorithm implemented in OpenCL is the direct bilateral filter with both the distance and range weight functions being arbitrary. The actual shape of the distribution is irrelevant in terms of speed, as both functions can be precomputed once at the start of the processing for necessary values, and used as a lookup table from the OpenCL constant memory, which is cached to allow fast access.

An OpenCL application was constructed in a way that there is a thread for each pixel, which computes Eq.(22) for a window centered at that pixel. Each thread has to access the surrounding \((2r+1)^2\) pixels around its assigned pixel in order to compute the output of the filter. As the amount of memory accesses is large and several threads must access the same pixels, it is reasonable to use an image object to store the input image instead of a traditional buffer. Image objects are spatially cached, i.e. memory accesses to the same area of the image do not have to be retrieved from expensive global memory, but are available in the device cache. Each thread loops through the sum in Eq.(22) reading and processing all the channels for a single pixel on each iteration.

After processing, the image can either be returned to the CPU for further processing, buffering, saving etc, or using OpenCL/OpenGL interoperability, sent directly to the display.

In our application, we used the bilateral filter as the main building block of the hypothesis filter for depth map deblacking. The bilateral filter is applied to filter a cost volume in order to find the optimal depth value for each pixel. The color and spatial weights are computed only one and then applied to the cost volume slices. Winners-take-all procedure across the slices selects the depth value.
5.3.3 Sliding Windowed DCT and VBM3D on OMAP

An implementation of the sliding windowed DCT on the OMAP was realized. It is following the same design paradigm of performing the tasks in parallel on the ARM and the DSP while sharing a common buffer between the two as the bilateral filter. The extraction of blocks and conversion from 8 to 16 bits per value is done on the ARM, while the DSP performs the DCT, thresholding and inverse DCT. Both DCT’s are done using TI’s Imglib library, which takes the input as 16 bit fixed point. The thresholded buffer is then returned to the ARM, which combines the blocks to form the final result of the filtering.

Because performing VBM3D on a frame includes the same operations as the SW-DCT, its performance is evaluated accordingly.

The test material used was a 480x272 YUV420 video sequence from the Mobile 3DTV project’s video library [54], which has been impaired with a low quality level DCT compression, resulting in very clear block boundaries, although the performance of the used techniques is not dependant on the content of the video.

The choice of color model between RGB and YUV (or CIE-Lab, as suggested by Tomasi [37]) has an effect on the computational performance of the filtering, depending on from which channels the weights are computed and to which channels they are applied. The ARM+DSP version was not extended to cover more than only the Y channel as it was already too slow for the purpose.

The OpenCL version uses all three RGB channels for calculating the weights and applies them to all three channels, so it gives in a sense the worst-case performance in terms of channel selection.

5.4 Experimental results

The sliding window DCT was measured to take approximately 0.90 seconds while processing a 320x240 grayscale image with 8x8 DCTs when the window was moving one pixel at a time. The amount of computation is directly proportional to the amount of pixels in the image, as one DCT-threshold-IDCT operation is done for each pixel. Therefore for a similarly sized image as before (480x272), the time would be 1.5 seconds per frame assuming only the Y component is considered. The performance is very sensitive to the step size of the sliding window, quickly becoming faster when it is increased, but it will also quickly start losing the overcompleteness which is responsible for the effect it produces.
To get an estimate about the resource demands of VBM3D we assume the availability of motion vectors after video decoding to facilitate the grouping step in the algorithm. Thus, VBM3D does not need to find the suitable matches, but can extract that information from e.g. motion vectors. In addition to the DCT computation done by the SW-DCT, it would still have to perform a 1D DCT for each stack of pixels in a stack of blocks centered on each pixel (8x8x480*272). Therefore it would take over 8 million DCTs and corresponding IDCTs to complete the transforms for a single frame. Even if doing an efficient 2D DCT took the same time as 8 horizontal and 8 vertical 1D DCTs. Those 8 million operations back and forth would take 4 times as long as the 2D DCT’s, making a total of over 7.5 seconds per frame.

5.4.2 Bilateral filter on OMAP: DSP+ARM

The final version of the ARM+DSP implementation processes one frame in 0.66 seconds. The computation of histograms for all of the window positions (centered on each pixel) takes approximately 250 ms on the ARM processor and the application of weights into the image 600 ms. The ARM side does not take advantage of NEON optimization, but as the bottleneck is the application of weights on the image and the operations are done in parallel, it has no effect to the overall processing time. The rest of the time, approx. 60ms is spent on moving content in memory etc. The total time can be expected to increase significantly if more than one channel is taken into account.

In comparison, Porikli achieved 0.06 seconds per frame on a desktop computer on a 1MB grayscale image with 16 bins [38]. This performance was fairly matched by running practically the same implementation on a PC as on the DSP, with the exception of using floating point arithmetic instead of fixed point. Further reduction in histogram bins was found visually disturbing due to the “comic book effect” with reduced color depth.

5.4.3 Bilateral and hypothesis filter on OpenCL

The key factor in the performance of the OpenCL application is the time it takes to transfer the data to the device, process it, and retrieve it back. Using an OpenCL specific profiler, it was determined that the 522kB of 32 bits per pixel image data takes 2.5ms to be transferred to or from the device, making the memory transfer take 5ms total. Processing time of the kernel is dependent on the window size.
more use the GPU gets from the caching properties of the image object. The results in Figure 65 are given for processing all three color channels.

For processing a single (depth) channel the performance gets even better. The NVIDIA Visual profiler returned an evaluation of the speed of the hypothesis filter of 43 ms/frame for filtering with block size of 9x9 pixels. This accounts for approximately 23 fps which is a good compromise between spatial improvement of visual quality and sufficient motion. Figure 66 illustrates the performance in terms of fps for different filter sizes for the ‘Bullinger’ sequence.

![Figure 66. Performance of Hypothesis filter for Bullinger (320 x 192) sequence for varying filter sizes](image)

An illustration of filtering results for the ‘Car’ sequence is given in Fig. 5.

### 5.5 Conclusions

The performance of the bilateral filter on the OMAP3530 did not reach the objective of processing stereo video in real time. While the implementation is likely not perfectly optimized, the key factors have been taken into account. Still, the speed is at least an order of magnitude slower than needed for a real time application. This suggests that the amount and type of processing power the OMAP3 platform offers is not suitable for the computation the bilateral filter requires. Furthermore, it would also not be possible to allocate all the resources of the platform to video post-processing, as the decoding and color space conversion also run at the same time. One can resort for implementing the filter in a specialized hardware, for which the current tests are quite instructive.

The OpenCL version looks very promising on terms of computational performance. Even when performing the filtering on all RGB channels, reasonable speeds are achieved. With this implementation, a window radius of approximately 4, i.e. 81 pixels in total would be feasible in a real time, full-colour stereo video application. Significant improvements are possible by code level optimization and taking advantage of all the properties the GPU hardware is offering. It is also only using the GPU, and leaving the CPU free for other tasks.

The bilateral filter is especially suitable for the parallel processing, as there are no dependencies between the processing of neighboring pixel values. With a properly configured thread distribution, the OpenCL image object shines with its ability to cache the pixel values according to their spatial location in the image, instead of their location in the memory. This corresponds more to the way the pixels are accessed while filtering.
Sliding window DCT is roughly in the same order of magnitude with the bilateral filter, so it is possible to reach similar performance with a well designed OpenCL implementation. In contrast, the VBM3D is very complex algorithm and its implementing for real-time applications should be based on specialized hardware.

Figure 67 ‘Car’ sequence, frame no. 90, (a) Original depth (b) Depth encoded with QP = 40 (c) Filtered with block size 9 (d) Filtered with block size 11 (e) Filtered with block size 13 (f) Filtered with block size 15
6 Conclusions

A deblocking and sharpening method has been proposed for improving the visual quality of stereo video. The filter utilizes the non-local collaborative filtering paradigm and suggests grouping of stereo volumes into 4-D structures to be filtered together. The grouping is very beneficial for improving the stereo correspondences and increasing the stereopsis, while the blocking effects are removed completely.

A novel occlusion filling algorithm for DIBR technique has been introduced, and a comparison with other techniques using objective and subjective tests has been performed. The algorithm implements the non-local inpainting approach in a computationally efficient way. The results show the superiority of the algorithm with respect to the other algorithms tested.

Several optimized implementations of filtering algorithms were realized and tested. The performance of the bilateral filter on the OMAP3530 did not reach the objective of processing stereo video in real time. OpenCL version of bilateral filter tested on a netbook shows promising results and could be used on that hardware for real time processing. The implementation of the hypothesis filter for depth deblocking utilizing this filter proved very efficient and running in real-time and returns good filtering results. Sliding window DCT is close to bilateral in performance, so the same conclusions apply to it. VBM3D is considered computationally too demanding for a DSP/GPU implementation. A specialized hardware for the next generation of mobile platforms should be targeted instead.

References


MOBILE3DTV - Mobile 3DTV Content Delivery Optimization over DVB-H System - is a three-year project which started in January 2008. The project is partly funded by the European Union 7th RTD Framework Programme in the context of the Information & Communication Technology (ICT) Cooperation Theme.

The main objective of MOBILE3DTV is to demonstrate the viability of the new technology of mobile 3DTV. The project develops a technology demonstration system for the creation and coding of 3D video content, its delivery over DVB-H and display on a mobile device, equipped with an auto-stereoscopic display.

The MOBILE3DTV consortium is formed by three universities, a public research institute and two SMEs from Finland, Germany, Turkey, and Bulgaria. Partners span diverse yet complementary expertise in the areas of 3D content creation and coding, error resilient transmission, user studies, visual quality enhancement and project management.

For further information about the project, please visit www.mobile3dtv.eu.

Tuotekehitys Oy Tamlink
Project coordinator
FINLAND

Tampereen Teknillinen Yliopisto
Visual quality enhancement, Scientific coordinator
FINLAND

Fraunhofer Gesellschaft zur Förderung der angewandten Forschung e.V
Stereo video content creation and coding
GERMANY

Technische Universität Ilmenau
Design and execution of subjective tests
GERMANY

Middle East Technical University
Error resilient transmission
TURKEY

MM Solutions Ltd.
Design of prototype terminal device
BULGARIA

MOBILE3DTV project has received funding from the European Community’s ICT programme in the context of the Seventh Framework Programme (FP7/2007-2011) under grant agreement nº 216503. This document reflects only the authors’ views and the Community or other project partners are not liable for any use that may be made of the information contained therein.